EXTENDING SECURITY OF PACKAGE MANAGER WITH DNS

Martin Sehnoutka
Master Degree Programme (2), FEEC BUT
E-mail: xsehno00@stud.feec.vutbr.cz

Supervised by: Jan Jeřábek
E-mail: jerabekj@feec.vutbr.cz

Abstract: This paper describes a method of public key verification based on DNS, that can be used to further increase security of package managers. On top of this method, an extended version is proposed, that can also protect repository metadata. An implementation for the RPM ecosystem is proposed corresponding to the theoretical background.

Keywords: Package management, OpenPGP, DNS, DNSSEC, RPM

1 INTRODUCTION

Package managers are an essential part of every modern Linux distribution. They are usually used to distribute software packages, e.g. Wireshark or Firefox, but in general they can be used to manage any kind of content. The primary method of package delivery is via the Internet, which is inherently insecure, therefore it is necessary to ensure package integrity. This is done by using signatures, as known from the public-key cryptography, and cryptographic hash functions [1]. But the public keys themselves must be obtained somehow and considered trusted before they can be used. In the rest of this paper, a method based on the Domain Name System (DNS) and applied to the RPM Package Manager (the acronym is recursive) will be described, that can be used to further increase security.

Nowadays, when a user of some RPM based distribution (e.g. Fedora or CentOS) wants to install a package for the first time, a series of steps precede the installation itself. First, the public key is downloaded from the Uniform Resource Locator (URL) provided in the repository configuration file. The key is usually a local file or a file available over the Hypertext Transfer Protocol Secure (HTTPS). Then the frontend package manager asks the user whether the key is trusted and this is the tricky part. The user must use a browser to find information about this key and compare the fingerprint manually. The situation is even worse in case of automatic package installation as done by Graphical User Interface (GUI) package managers or automation systems, e.g. Ansible. These systems skip this step altogether as there is currently no way for them to do otherwise. Finally, the key is imported and trusted forever. But again this is a problem because there is no automatic way to inform the package manager, that the key was revoked. The information must be delivered to the user, and he must manually remove the key.

2 STRUCTURE OF AN RPM PACKAGES REPOSITORY

Before the proposed extension can be further explained, an RPM repository needs to be described. It is a server that contains repository metadata and a set of packages. The metadata is stored in 2 files: *repomd.xml* and *primary.xml*. They contain hashes of the provided packages so that they can be tested for corruption during transmission, but it cannot be used to verify authenticity as anyone can create these hashes. The metadata files can be signed but they usually are not unlike in other distributions like Debian or Gentoo, instead RPM includes signatures in every package so that each of them can be verified separately.

3 EXTENDING THE CURRENT SYSTEM WITH DNS

Based on the theoretical background introduced in the previous sections, two ways are proposed in which the current package distribution system can be extended in order to prevent certain types of attacks. The first version, which is called "*minimal*" is supposed to require only minimal changes to the RPM ecosystem and it can prevent Man in the middle (MITM) attacks during the key import phase as well as it can help with key revocation and replacement after a key leakage incident occurres. The second version, called "*extended*", builds on top of the minimal version and it can also prevent so-called freeze attack [1], where the attacker restrains users from installing security updates by providing only outdated packages.

3.1 MINIMAL VERSION

Both versions can be divided into server and client part. The server side represents a packager, a person, a community or in general any entity, that creates packages, signs them and publishes them on a domain (e.g. example.com). An assumption is made, that this entity owns this domain and therefore it can publish new Resource Records (RR) in the authoritative DNS server for this domain. This entity is represented by a Pretty Good Privacy (PGP) key pair, that is used for signing of the packages. PGP keys are tied with an email address (e.g. packager@example.com), which belongs to this domain as well. Conveniently, an RR type for PGP keys has already been standardized [2], so the OPENPGPKEY RR type can be used to publish the public key in the DNS system. Finally, the DNS system itself does not guarantee authenticity of its messages, in order to verify obtained RRs, the Domain Name System Security Extensions (DNSSEC) must be used.

The client side is implemented in the Dandified Packaging Tool (dnf). When the client wants to install a package from this repository for the first time, dnf downloads the PGP key from the web server using HTTPS and now it can automatically verify this key using an implementation of a verifying resolver (e.g. libundound), that downloads the key RR and verifies RR signatures from the key itself up to the root zone. This is known as the DNSSEC chain of trust.

The second application, key revocation, is done by simply replacing the original key with a new one. Whenever dnf wants to install a new package (this applies to updates as well, because an update is basically installation of a new version), it first verifies all the keys and if there is some discrepancy, it can check the web server for a new version of the verification key or it can just reject to install this package.

3.2 EXTENDED VERSION

The DNS with DNSSEC can be utilized to further improve security of the whole package distribution system especially in cases, where plain HTTP connection is used for communication with a repository. As mentioned earlier, the freeze attack can be used against package managers. This attack vector consists of MITM where the attacker provides old, but correctly signed packages and waits for new security vulnerabilities to occur in installed packages (e.g. httpd).

In order to prevent this attack, a hash of the repository metadata and a timestamp can be included in the DNS database. Of course, the timestamp has a short-period timeout, so that the RR cannot be frozen as well. Also, the repository metadata can be signed. This is already supported by RPM, but not widely used in Fedora. It works simply by creating a detached signature using GPG. The key used for this purpose can be and should be different from the key used for signing packages. This structure together with the package verification key resembles The Update Framework (TUF) as proposed in [1] with the difference that it is implemented using DNS. The whole idea of the extended version is captured in Figure 1. The yellow part is the RPM ecosystem that will not be modified, the blue part will be used in the minimal version and the red part is where the extended version is applied.

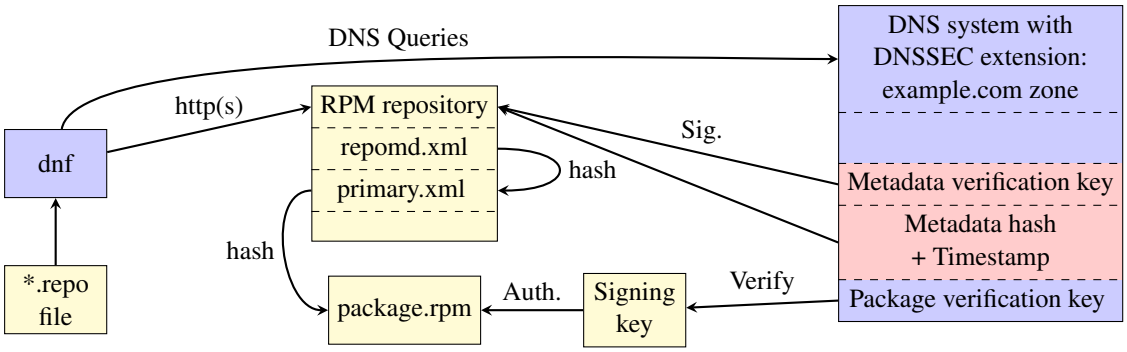

Figure 1: Structure of data stored inside a DNS server,its relation to RPM repository and interaction with dnf

4 IMPLEMENTATION

A proof of concept implementation has been written using the Python language. The libunbound library was used as a resolver implementation with Python API and so was the python-gnupg library, which is a Python wrapper around the GPG tool. A virtual environment was created with a set of DNS servers, that resembles the global DNS system, but it is possible to control it and create correct and broken configurations, so that the implementation is properly tested.

5 CONCLUSION

In this paper, a way was described in which dnf can utilize DNS in order to increase security of package management. Once the implementation is ready for production use, it will be included in the upstream dnf and tested by Fedora users. If the system proves to be useful it can be beneficial especially in cloud and desktop environment, where packages are installed non-interactively.

REFERENCES

[1] SAMUEL, Justin, Nick MATHEWSON, Justin CAPPOS a Roger DINGLEDINE. Survivable key compromise in software update systems. In: *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*. New York, New York, USA: ACM Press, 2010, s. 61-72. DOI: 10.1145/1866307.1866315. ISBN 9781450302456. Available from: http://portal.acm.org/citation.cfm?doid=1866307.1866315

[2] RFC 7929: DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP. *RFC online* [online]. California: Internet Engineering Task Force (IETF), 2016 [cit. 2017-11-07]. Available from: https://tools.ietf.org/html/rfc7929